Sistema de votació electrònica sobre corbes el·líptiques amb mescla de vots mitjançant Random Group Full Checking

Treball de Final de Carrera

Enginyeria en Informàtica - Escola Politècnica Superior

Universitat de Lleida

Autor:

Oriol Carro i Vidallet

Directors:

Josep M. Miret Biosca i Francesc Sebé Feixas

Juliol de 2011

Índex

Llista d'algoritmes

Capítol 1

Introducció

L'objectiu del projecte és implementar un sistema de simulació de votació electrònica, emprant una adaptació sobre corbes el·líptiques del criptosistema ElGamal, per tal d'estudiar-ne la seva viabilitat, centrant l'atenció en temes de seguretat, especialment en el procés de mescla de vots per tal de desvincular un vot de la persona que l'ha emès.

1.1 Context

Actualment, podem realitzar molts tràmits a través d'Internet, ja siguin oficials o no. Els avantatges d'aquesta opció són la comoditat, la immediatesa i la possibilitat de dur a terme tasques que presencialment seria impossible fer-les. Tanmateix, la utilització d'aquestes eines encara desperta un cert temor al públic a qui van dirigides ja que s'ha de confiar en què tot el procés es desenvolupa de forma segura. La finalitat d'aquest treball és demostrar que es pot afegir la votació electrònica al conjunt de tràmits que es poden fer telemàticament amb seguretat. Aquest treball s'emmarca dins d'un projecte de votació electrònica desenvolupat per la Universitat de Lleida en col·laboració amb l'empresa *Scytl* mitjançant un projecte *Avanza* concedit pel *Ministerio de Industria*.

1.2 Estructura del projecte

Podem distingir quatre blocs principals:

- **Votació electrònica**: s'exposen els conceptes bàsics, incidint sobretot, en els requisits que ha de complir qualsevol procés electoral perquè aquest sigui segur i en les tècniques criptogràfiques que disposem per fer-ho.
- **Criptografia amb corbes el·líptiques**: en aquest capítol s'introdueix què és una corba el·líptica i les operacions definides sobre aquestes (suma de punts i multiplicació d'un punt per un escalar). A partir de les operacions s'enuncia el planteig sobre corbes el·líptiques del problema del logaritme discret, que ens servirà per definir el criptosistema ElGamal el·líptic.
- **Sistema de votació electrònica emprant ElGamal el·líptic**: un cop hem vist les eines criptogràfiques necessàries s'ha d'implementar l'aplicació. Per fer-ho, cal conèixer quin ha de ser el seu comportament, és a dir, quines etapes el composen i com modelitzar-les amb les tècniques anteriors. Per deixar més clars els conceptes exposats es presenta un exemple simulant tot el procés d'una votació electrònica.
- **Resultats i conclusions**: en aquest darrer capítol es mostren els resultats obtinguts de les proves realitzades i se n'extreuen les conclusions del treball. També es dóna una orientació sobre quin pot ser el següent pas per millorar el sistema.

A part d'aquestes seccions, també hi ha dos apèndix:

A. **Descripció i implementació de les proves de correctesa**: a més de comprovar que la mescla s'ha fet de forma correcta, també cal fer alguna altra verificació. És en aquest apartat on s'exposa i s'implementa com es valida que un vot correspon a un candidat i que el desxifratge s'ha realitzat correctament.

B. **Implementació del sistema**: en aquest darrer capítol es presenta el codi, en Java, de les funcionalitats més interessants del sistema.

1.3 Eines emprades

Per tal de desenvolupar l'aplicació s'ha escollit el llenguatge de programació Java. En un primer moment es van mirar les classes que oferia aquest llenguatge per treballar amb corbes el·líptiques i diferents llibreries, però es va decidir implementar les operacions sobre corbes el·líptiques, tal i com es defineixen en l'apartat 3.1 de nou, ja que ens oferia més flexibilitat.

Els càlculs dels tres exemples proposats s'han dut a terme amb el calculador simbòlic *Sage* [13], que ens permet treballar de forma senzilla i ràpida amb corbes el·líptiques.

Capítol 2

Votació electrònica

Una elecció és un procés de presa de decisions a través del qual un conjunt de persones (definides en el cens) expressen, de forma secreta, la seva opinió, escollint l'opció desitjada entre totes aquelles disponibles (candidatures). El terme votació és molt genèric, ja que existeix una gran diversitat d'eleccions, en funció de diversos factors, com ara: el cens, l'àmbit en què es realitza, l'oficialitat, el nombre de canditatures, si es pot definir preferència entre les opcions, la finalitat,... En qualsevol procés electoral, trobem els següents agents:

- **Autoritat**: persona o grup de persones encarregats de l'elecció. Es responsabilitza de definir els paràmetres del procés, els requisits dels votants, els objectius, etc.
- **Votant**: qualsevol persona amb dret a participar en la votació.

Quan parlem de votació electrònica ens referim a automatizar alguna de les fases del procés, especificades en el següent apartat.

2.1 Etapes

Les etapes bàsiques d'una votació electrònica són:

- Previ a la votació:
  - Arran de la voluntat de realitzar una votació, cal anunciar-la, descrivint-ne les seves característiques.
  - Registre dels votants. S'ha d'obtenir la informació dels votants per tal de poder-los identificar posteriorment.
- Durant la votació
  - Autenticació del votant, en funció de les dades registrades anteriorment.
  - Selecció dels candidats.
  - Enviament del vot xifrat des de la màquina client al servidor.
  - Validació del vot: es comprova que el votant hagi votat un sol cop i a un nombre de candidats permès.
- Posterior a la votació
  - Transferència dels vots des del servidor cap a l'autoritat encarregada de fer l'escrutini.
  - Permutació dels vots per tal de desvincular la informació d'un vot del votant que l'ha emès.
  - Desxifratge dels vots.
  - Escrutini
  - Auditoria
  - Publicació dels resultats.

2.2 Tipologia

En funció de quina de les fases anteriors automatitzem, podem diferenciar els següents tipus de votació electrònica [8]:

- **Reconeixement òptic de marques** (OMR): el votant ha d'exercir el seu dret a vot presencialment, emprant unes butlletes especials que mitjançant un

dispositiu d'escaneig poden ser llegides ràpidament. En aquest cas, automatizem l'escrutini.

- **Registre electrònic directe** (DRE): també es tracta d'una votació presencial en què al col·legi trobem un terminal que permet votar. La màquina ho emmagatzema en format electrònic. En aquest cas, automatitzem tot el procés realitzat durant la votació i el posterior a aquesta.
- **Vot electrònic remot**: el seu objectiu és evitar que s'hagi de votar presencialment, per tant, s'automatitzen totes les fases.

A partir d'ara, quan parlem de votació electrònica ens referirem a una votació electrònica remota.

2.3 Requisits de seguretat

Per tal de garantir la correctesa i validesa d'una elecció realitzada mitjançant un sistema de vot electrònic remot, s'han de complir una sèrie de propietats de seguretat [11]:

- **Autenticació**: només els votants autoritzats i registrats abans de la votació, és a dir, els que es troben en el cens electoral poden exercir aquest dret.
- **Unicitat**: cada votant present en el cens, només pot votar un cop.
- **Integritat**: qualsevol intent de manipulació del resultat ha de ser detectat pel sistema.
- **Privacitat**: no hi pot haver cap informació que relacioni una butlleta amb el votant que l'ha seleccionada.
- **Verificabilitat**: tot el procés ha de ser universalment verificable, és a dir, qualevol persona ha de poder comprovar-ne la correctesa.
- **No coercitiu**: un votant no es pot veure obligat per una altra persona a votar un candidat determinat, és a dir, no ha de poder demostrar a ningú el seu vot.

- **No informació**: mentre dura la votació, no es poden anar publicant resultats parcials.

En les eleccions tradicionals es pot anar supervisant el procés de votació a mesura que aquest es va desenvolupant, però això no succeix així en les electròniques. Les possibles alternatives són:

- Permetre als observadors instal·lar programes per tal de monitoritzar tot el procés. El problema, en aquest cas, rau en el fet que algú hauria de poder supervisar als supervisors i això ens porta a un bucle sense fi.

- Monitoritzar el *log* generat durant l'execució. El contrapunt d'aquesta solució és que es podria modificar el *log* de tal manera que no es pogués detectar que s'ha variat el contingut. També pot comprometre la privacitat del votant.

- Verificabilitat universal: es tracta de proporcionar els mitjans als auditors i observadors per verificar la correctesa del desxifratge dels vots, emprant proves criptogràfiques generades en el moment de desxifratge.

2.4 Paradigmes

Per tal de garantir les propietats de seguretat que ha de complir una votació, disposem de certes tècniques criptogràfiques. Les més emprades són la firma cega, el xifrat homomòrfic i els mixnets [4].

2.4.1 Firma cega

És complex verificar l'autenticació del votant i garantir la seva privacitat, és a dir, evitar que se'l relacioni amb el vot emès. Aquest mecanisme proposa que una **autoritat d'identificació** firmi el vot xifrat, així, s'emet el vot un cop autenticat, protegint la identitat del votant [8]:

Votant	**Autenticador**	**Servidor**
Es xifra el vot V amb la clau pública		
Factor de ceguesa: r $c = V \cdot r$		
Credencials: id $M = (c, id)$		
$\xrightarrow{\text{S'envia } M}$		
	Valida id	
	$M' = M$ signat amb la clau privada	
$\xleftarrow{\text{S'envia } M'}$		
Verifica la firma		
Inverteix r $V' = M' \cdot r^{-1}$		
$\xrightarrow{\text{S'envia } M'}$		
		Es verifica la firma i es guarda el vot

Per evitar que es firmin missatges no vàlids, s'utilitzen tècniques de "tall i elecció": el votant envia a l'autenticador p missatges xifrats i cegs. L'autenticador n'escull $p-1$ i en demana els factors de ceguesa.

2.4.2 Xifratge homomòrfic

La idea d'un xifratge homomòrfic és poder treballar amb un conjunt de vots de forma col·lectiva, enlloc de fer-ho vot a vot.

Genèricament, siguin v_1 i v_2 dos vots qualsevols i ϕ i θ dues operacions. Es diu que l'operació té la propietat homomòrfica si satisfà:

$$E(v_1)\ \phi\ E(v_2) = E(v_1\ \theta\ v_2).$$

En funció de quines siguin les operacions ϕ i θ podem distingir diferents tipus d'homomorfisme:

- **Additiu**: el resultat d'operar els vots individuals xifrats es correspon al xifratge de la suma dels vots en clar. Per tant, en desxifrar el resultat, obtenim la suma dels vots. Criptosistemes amb aquesta propietat: ElGamal exponencial i Paillier [5].
- **Multiplicatiu**: el resultat d'operar els vots individuals xifrats es correspon al xifratge de la multiplicació dels vots en clar. Per tant, en desxifrar, obtenim la multiplicació dels vots. Criptosistemes amb aquesta propietat: ElGamal i RSA.

2.4.3 Mixnets

L'objectiu d'aquesta tècnica és realitzar una permutació dels vots per tal de deslligar el vincle que s'estableix entre un vot i el votant que l'ha emès. El projecte se centra en aquesta tècnica, per tant, el procediment de la mescla de vots serà explicat en més detall més endavant.

Aquestes eines no són excloients, és a dir, es poden combinar i utilitzar-ne més d'una. Per exemple, en el nostre cas, es barrejaran un conjunt de vots que haurem empaquetat prèviament gràcies a que l'adapatció del criptosistema d'ElGamal per simular una votació compleix la propietat d'homomorfisme explicada a 2.4.2.

2.5 Avantatges i inconvenients

La votació electrònica ens aporta força avantatges. En primera instància, el més visible és el fet de votar remotament, és a dir, no cal anar a un col·legi electoral. A part de la comoditat que això comporta, també s'ha valorar que permet que una votació sigui més accessible per persones amb alguna discapacitat.

A nivell més intern de la votació, permet un control d'errors dels vots, o sigui, es pot detectar un vot invàlid abans que aquest sigui emmagatzemant al servidor; si es tracta d'un error del votant, se li oferirà l'opció de refer-lo, evitant així vots nuls involuntaris. Cal destacar, positivament, que es permet una verficació universal de tot el procés.

Un altre avantatge del vot electrònic és que el recompte de vots és molt ràpid. Un cop acabat el termini per votar, s'obtenen els resultats en poc temps.

Com a inconvenient, tenim el tema de la seguretat, és a dir, la dificultat d'implementar el sistema de forma que es compleixin tots els requisits citats en l'apartat 2.3. De totes formes, més que com un inconvenient es pot veure com una oportunitat, ja que el fet de complir els requisits ens aporta una votació electrònica molt segura i permet, a mesura que es van descobrint nous atacs a les tècniques emprades, investigar i crear-ne d'altres de més resistents i eficients.

Capítol 3

Criptografia amb corbes el·líptiques

En els darrers anys, la criptografia amb corbes el·líptiques està adquirint gran importància, sobretot aquells criptosistemes que es basen en el problema del logaritme discret el·líptic (EC-DLP) ja que garanteixen la mateixa seguretat que els construïts sobre un grup multiplicatiu d'un cos finit primer, però treballant amb claus de longitud molt més petita (veure taula 3.1), el que comporta un cost computacional força inferior [7].

DLP en $\mathbb{F}_p^*$ (bits)	**DLP el·líptic (bits)**
1024	163
3072	256
7680	384
15360	512

Taula 3.1: Comparativa del tamany de les claus del DLP i el EC-DLP per un mateix nivell de seguretat

Definició 1 (Problema del logaritme discret)**.** *Donat un grup cíclic finit G, un generador g de G i un element x de G, trobar un enter n tal que $x = g^n$, és a dir, el logaritme discret de x en base g.*

Aquest és un problema computacionalment difícil sobre el que basa la seva seguretat el criptosistema ElGamal.

3.1 Corbes el·líptiques

Definició 2 (Corba el·líptica)**.** *Una corba el·líptica sobre un cos* $\mathbb{K}$ *és una corba algebraica sense punts singulars que ve donada per l'equació de Weierstrass:*

$$E/\mathbb{K} : y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6, \quad a_i \in \mathbb{K}.$$

Si la característica del cos és diferent a 2 i 3, es pot expressar amb l'*equació de Weierstrass reduïda*:

$$E/\mathbb{K} : y^2 = x^3 + ax + b, \quad a, b \in \mathbb{K},$$

amb discriminant:

$$\Delta = -16(4a^3 + 27b^2) \neq 0.$$

Donada una corba el·líptica sobre un cos $\mathbb{K}$, denotem per $E(\mathbb{K})$ el conjunt de punts $P = (x, y) \in \mathbb{K} \times \mathbb{K}$ que satisfan l'equació de la corba més el punt de l'infinit $\mathcal{O}$.

3.1.1 Suma de punts

L'operació suma de dos punts que pertanyen a una corba el·líptica es defineix a partir del mètode de la corda i la tangent [7]. Analíticament, les coordenades del punt suma s'expressen de la forma següent:

Definició 3 (Suma de punts)**.** *Siguin* $P = (x_1, y_1)$ *i* $Q = (x_2, y_2)$ *dos punts de la corba el·líptica* $E/\mathbb{K} : y^2 = x^3 + ax + b$*.*

Aleshores, les coordenades del punt $P + Q$ *són:*

$$P + Q = (\lambda^2 - x_1 - x_2, (x_1 - x_3)\lambda - y_1),$$

$$on\ \lambda = \begin{cases} \dfrac{y_1 - y_2}{x_1 - x_2} & si\ x_1 \neq x_2, \\ \\ \dfrac{3x^2 + a}{2y_1} & si\ x_1 = x_2, y_1 \neq -y_2. \end{cases}$$

3.1.2 Multiplicació d'un punt per un escalar

Una altra operació que podem realitzar amb un punt de la corba el·líptica és multiplicar-lo per un escalar k. L'operació multiplicació es defineix a partir de l'operació suma:

Definició 4 (Multiplicació d'un punt per un enter)**.** *Sigui P un punt de la corba el·líptica i k un enter. Aleshores,*

$$k \cdot P = \begin{cases} P + ...^{k\ vegades} + P, & si\ k > 0 \\ \mathcal{O}, & si\ k = 0 \\ (-P) + ...^{k\ vegades} + (-P), & si\ k < 0 \end{cases}$$

És un càlcul senzill i ràpid de fer. Existeixen diferents algoritmes, però el més utilitzat és el mètode binari [1]. La idea bàsica d'aquest algoritme és expressar el nombre k en binari, i multiplicar el punt per 2 si el dígit és un '0' i multiplicar per dos i sumar el punt si és un '1'. Així doncs, tenim que el nombre d'operacions a realitzar és $\log_2(k)$. Per exemple, si $k = 13 \rightarrow 1101$ tenim:

$$k \cdot P = 13 \cdot P = 2\ (2\ (2 \cdot P + P)\ ) + P$$

Aquesta és l'operació que s'utilitza en la criptografia de corbes el·líptiques.

3.1.3 Corbes el·líptiques sobre un cos finit $\mathbb{F}_p$

Des de la vessant criptogràfica, ens interessa treballar amb corbes el·líptiques definides sobre un cos finit primer $\mathbb{F}_p$ perquè el problema del logaritme discret és difícil de resoldre.

L'interès de les corbes definides sobre un cos finit $\mathbb{F}_p$ es troba en el nombre de punts, el seu cardinal, que compleix el teorema de Hasse [7].

Teorema 3.1 (Hasse)
El cardinal $m = \#E(\mathbb{F}_p)$ satisfà:

$$p + 1 - 2\sqrt{p} \leq m \leq p + 1 + 2\sqrt{p}$$

Per tal d'evitar possibles atacs al problema del logartime discret el·líptic, com ara el de Pohlig-Hellman, ens interessa que el cardinal sigui de la forma $m = n \cdot h$ on n és un primer gran i h un cofactor petit (1 o 2).

3.1.4 EC-DLP

Definició 5 (Problema del logaritme discret el·líptic)**.** *Donat dos punts P i Q d'una corba el·líptica E definida sobre un cos finit $\mathbb{F}_p$ tal que $P = k \cdot Q$ trobar el valor de k.*

Existeixen algoritmes eficients per calcular el producte d'un punt per un enter, però en canvi, resoldre el problema invers (EC-DLP) és díficil. És aquesta dificultat la que aprofitarem per tal d'implementar els criptosistemes el·líptics.

3.2 ElGamal el·líptic

Es tracta d'una adaptació del criptosistema ElGamal definit sobre el grup multiplicatiu d'un cos finit per tal de definir-lo sobre el grup de punts d'una corba el·líptica. L'avantatge que ens aporta, com ja s'ha comentat anteriorment és que podem treballar amb claus de longitud força més petita, de forma que els càlculs a realitzar tarden menys temps. La idea bàsica per fer l'adaptació a corbes el·líptiques és transformar les multiplicacions i exponenciacions en un grup multiplicatiu $\mathbb{F}_p^*$ en sumes i multiplicacions, respectivament, en la versió el·líptica.

Sigui $E_{ab}(\mathbb{F}_p)$ una corba el·líptica que compleixi que $\#E(\mathbb{F}_p) = n \cdot h$ on n és un primer gran i el cofactor h és 1 o 2.

Sigui M el missatge que es vol transmetre.

El funcionament del criptosistema d'ElGamal el·líptic és el següent:

Paràmetres: (p, a, b, P, n, h) tal que $\#E_{ab}(\mathbb{F}_p) = n \cdot h$

Clau privada: $d \in [1, n-1]$

Clau pública: $Q = d \cdot P$

Xifratge: $M \to (r \cdot P, M + r \cdot Q)$ on r és aleatori

Desxifratge: $(r \cdot P, M + r \cdot Q) \to (M + r \cdot Q) - d(r \cdot P) = M + r \cdot Q - rd \cdot P = M$

Aquest criptosistema és una adaptació per a corbes el·líptiques del criptosistema exposat a [6].

Capítol 4

Sistema de votació electrònica emprant EC-ElGamal

En aquest capítol se simula el procés d'una votació electrònica que usa el criptosistema ElGamal sobre corbes el·líptiques. En els primers apartats s'explica quines etapes la composen i com les podem modelitzar. A continuació s'aprofundeix més en la fase de la barreja de vots. Amb l'objectiu de facilitar l'assimilació dels contiguts exposats, també es presenta un exemple explicant pas a pas tot el procés electoral.

Per tal d'aconseguir millorar el rendiment del sistema, enlloc de treballar amb tots els vots de forma individual, aquests s'empaqueten, és a dir, els vots un cop xifrats passen a formar part d'un paquet, i a partir d'aquest moment tot el procés es desenvolupa a nivell de paquet. Això ens permet disminuir el nombre de desxifratges. Tanmateix, s'ha de controlar el nombre de vots emmagatzemats en aquests, ja que un cop desxifrats s'haurà de resoldre el problema de la motxilla per obtenir els vots individuals.

4.1 Etapes

Les etapes en què podem dividir el sistema són les següents:

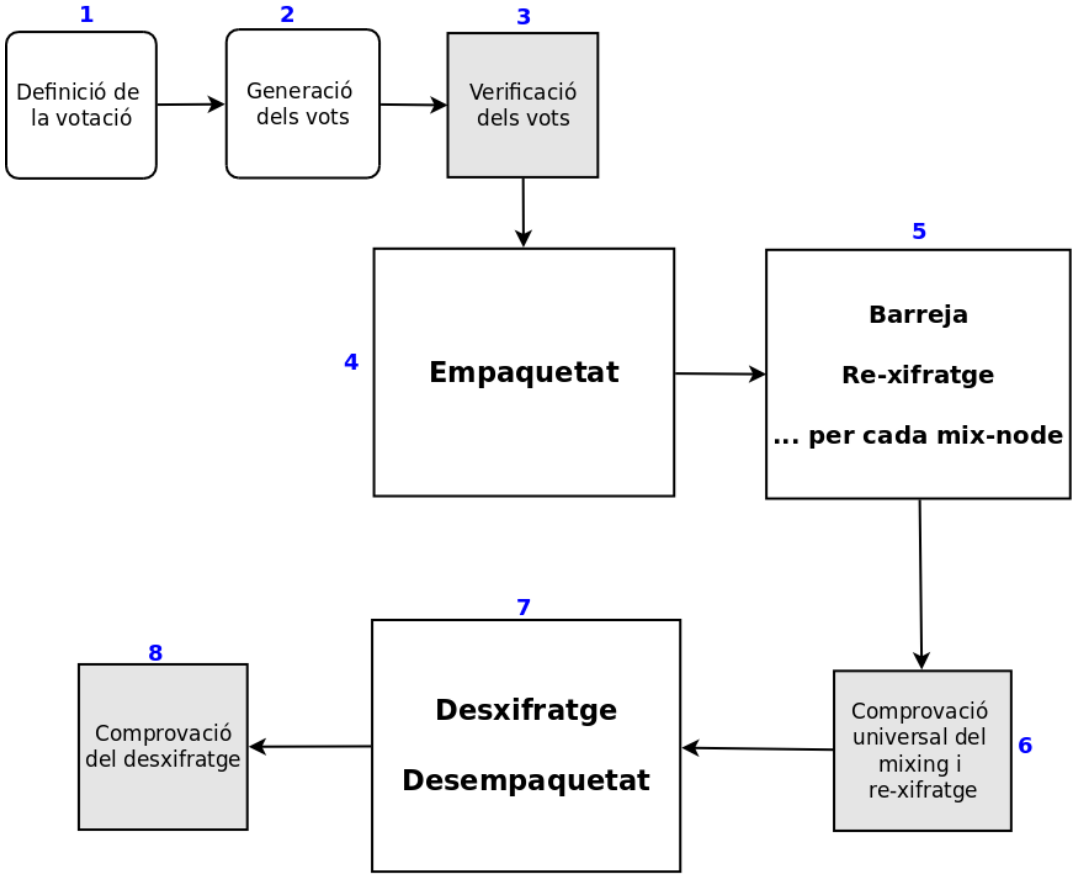

Figura 4.1: Etapes de la simulació

1. **Definició de la votació**. Podem separar aquesta etapa en diverses parts:

   - *Paràmetres de la votació*: tipus de criptosistema emprat, la clau pública per xifrar els vots, el cens, el nombre de candidats i el nombre de vots per persona.
   - *Paràmetres de la simulació*: nombre de vots per paquet, nombre de nodes del mixing i l'especificació de les verificacions que es vol realitzar.
   - *Informació addicional*: data, descripció,...

   En el cas que el criptosistema emprat sigui ElGamal el·líptic, també cal definir:

   - Corba el·líptica emprada: un primer p que determina el cos $\mathbb{F}_p$ on es

defineix la corba, els seus coeficients, un generador P dels punts de la corba i el seu cardinal.

- Clau privada: $d \in [1, n-1]$.
- Clau pública: $Q = d \cdot P$.

Això es defineix en diferents fitxers *xml*, permetent-nos d'aquesta forma canviar els paràmetres de la simulació sense haver de modificar el codi de l'aplicació. Els valors utilitzats en la implementació són:

Listing 4.1: Definició de la votació

```
<eleccio>
  <tipus> ECElGamal </tipus>

  <ParamatresGenerics>
    <cens> 50 </cens>
    <votsPersona> 1 </votsPersona>
    <nCandidats> 4 </nCandidats>
  </ParamatresGenerics>

  <ParametresECElGamal>
    <nMixNodes> 4 </nMixNodes>
    <tamanyPaquet> 5 </tamanyPaquet>
    <verificacioMixing> true </verificacioMixing>
    <verificacioDesxifrat> true </verificacioDesxifrat>
    <verificacioVot> true </verificacioVot>
  </ParametresECElGamal>

  <infoAddicional>
    <Data> 10-10-10 </Data>
    <Descripcio> Simulacio d'un proces electoral </Descripcio>
  </infoAddicional>
</eleccio>

<EC>
  <p> 44516852250937147720845982735484 27 </p>
  <a> 44516852250937147720845982735484 24 </a>
  <b> 20611183968086532029029961663885 14 </b>
  <gen>
```

```
29        <x> 188281465057972534892223778713752 </x>
30        <y> 341987549103317082716786189608268 8 </y>
31      </gen>
32      <n> 445168522509371477649189154254893 3 </n>
33      <h>1</h>
34    </EC>
35
36    <Q>
37      <Qx> 160098459156496762958131973543860 4 </Qx>
38      <Qy> 301296087973443783335623243076495 7 </Qy>
39    </Q>
```

2. **Generació dels vots**. Es generen els vots xifrats de forma aleatòria. La quantitat de vots generats és igual al cens, és a dir, entenem que tots els possibles votants exerceixen el seu dret. Un vot xifrat es representa amb dos punts de la corba el·líptica.

   Sigui $PesCandidat$ la llista que conté els punts de la corba que representen a cada candidat. L'algoritme de generació d'un vot és el següent:

   Algoritme 1 Generació d'un vot xifrat

   Entrada: $PesCandidat$

   Sortida: vot xifrat per un candidat aleatori

   $a \leftarrow$ enter aleatori $\in [0, numCandidats - 1]$

   $r \leftarrow$ enter gran aleatori

   Vot $\leftarrow$ xifrar$(PesCandidat[a], r)$

   retornar Vot

   Sigui K el punt de la corba el·líptica que representa al candidat a votar.

   Siguin P un generador del grup de punts de la corba i Q la clau pública.

   Sigui r l'enter gran amb el qual es vol xifrar el vot.

   Aleshores, l'operació de xifratge ha estat definida de la següent forma a 3.2:

Punt 1: $r \cdot P$

Punt 2: $K + r \cdot Q$

on r és un enter aleatori.

L'algoritme de xifratge d'un vot és:

Algoritme 2 Xifratge d'un vot

Entrada: K, r, E corba el·líptica sobre $\mathbb{F}_p$, P generador de E, Q clau pública

Sortida: vot xifrat pel candidat especificat a l'entrada

$P_1 \leftarrow r \cdot P$

$P_2 \leftarrow K + r \cdot Q$

Vot $= (P_1, P_2)$

retornar Vot

3. **Verificació dels vots**. Cal comprovar que els vots generats representen realment un candidat. En la simulació és evident que sí, donat que els hem generat nosaltres mateixos, però en una implementació real s'ha de fer i per tant s'implementa per calcular els temps de fer aquesta comprovació. Això es fa mitjançant un prova de coneixement nul. Es tracta d'una adaptació de la prova en el ElGamal multiplicatiu, exposada en [9], a corbes el·líptiques.

   L'explicació de la prova i el seu codi es troba en l'apartat A.1 de l'apèndix.

4. **Empaquetat dels vots**. Un cop hem verificat que els vots són correctes, aquests es van emmagatzemant en paquets de la mida que s'ha especificat en la definició.

   Cal destacar que és possible realitzar l'empaquetat de vots gràcies a la propietat homomòrfica. Com que el missatge que es vol enviar és un punt que representa un candidat, el criptosistema ElGamal el·líptic compleix la propietat homomòrfica. Per comprovar que és un criptosistema homomòrfic, siguin v_1, v_2,..., v_n el conjunt de vots d'un paquet, E l'operació d'encriptar i ϕ i θ dues operacions, per la definició donada a 2.4.2, s'ha de complir:

$$E(v_1)\ \phi\ E(v_2)\ \phi\ ...\ \phi\ E(v_n) = E(v_1\ \theta\ v_2\ \theta\ ...\ \theta\ v_n)$$

en el nostre cas, on ϕ i θ se substitueix per l'operació suma:

$$E(v_1)\ +\ E(v_2)\ +\ ...\ +\ E(v_n) = E(v_1\ +\ v_2\ +\ ...\ +\ v_n)$$

com que:

$$[(r_1 \cdot P)\ ,\ (v_1 + r_1 \cdot Q)] + ... + [(r_n \cdot P)\ ,\ (v_n + r_n \cdot Q)] =$$

$$= [((r_1 + ... + r_n) \cdot P)\ ,\ (v_1 + ... + v_n + (r_1 + ... + r_n) \cdot Q)]$$

es demostra la validesa de la propietat.

Això ens permet reduir la quantitat de desxifratges que s'hauran de fer, però també hem de tenir en compte que com més gran sigui el nombre de vots empaquetats, més costós serà resoldre el problema de la motxilla un cop desxifrat el conjunt de vots. Així doncs, s'haurà de buscar el terme mig, que faci més eficaç tot el procés.

Algoritme 3 Empaquetat d'un vot

Entrada: Vot xifrat *vot*

Sortida: Paquet *pq*

```
pq ← paquet actual
si pq està ple llavors
    pq ← crear nou paquet
fi si
pq ← pq.afegir(vot)
```

5. **Mixing**. Es barregen i re-xifren els paquets de vots per tal de desvincular un votant del seu vot. Es tracta d'una adpatació per a corbes el·líptiques de [3]. S'explica de forma més detallada en l'apartat 4.3.

6. **Verificació del mixing**. Com s'han modificat els paquets, s'ha de comprovar que el procediment ha estat correcte i no s'ha manipulat el contingut de cap vot. En la secció 4.3.1 s'exposa el seu funcionament.

7. **Desxifratge i desempaquetament**. Un cop s'ha acabat la votació, s'han de desxifrar tots els paquets i recuperar els vots individuals que hi havien emmagatzemats en aquest (problema de la motxilla).

Algoritme 4 Desxifratge i desempaquetament

Entrada: booleà CP, paquet de vots *pack*
Sortida: *resultat*

```
m ← desxifrar(pack)
si CP == cert llavors
    CPProver ← provador del desxifratge
    CPVerifier ← verificador del desxifratge
    bool v ← CPProver.prover(m, pack)
    si v == cert llavors
        resultat ← resoldre motxilla
        retornar resultat
    sinó
        retornar null
    fi si
sinó
    resultat ← resoldre motxilla
    retornar resultat
fi si
```

La implementació de la motxilla es troba en l'apartat B.2 de l'apèndix.

8. **Verificació del desxifrat**. Per comprovar la correctesa del desxifratge es realitza la prova Chaum-Pedersen.

   En l'algoritme presentat en el punt anterior, veiem que es rep com a entrada un booleà CP amb l'objectiu d'indicar si volem que es realitzi o no la verificació del desxifratge.

   La descripció i la implementació de la prova es troba en l'apartat A.2 de l'apèndix.

4.2 Modelització

Per tal de modelitzar una votació a partir del criptosistema ElGamal el·líptic, seguim els següents passos:

1. Assignem un punt a cada candidat. Per tal de no tenir ambigüitats a l'hora de desxifrar, determinem per cada candidat i cada possible nombre de vots obtinguts (per paquet) un punt de la corba, de forma que al sumar el punts que representen a cada candidat, el resultat sigui un punt diferent per cada possible combinació.

   Sigui n el número màxim de vots per candidat per paquet i sigui el punt P un generador del grup de punts de la corba, obtenim la taula 4.1.

	1 Vot	**2 Vots**	...	**n Vots**
$\mathbf{P_1}$	$1 \cdot P$	$2 \cdot P$	...	$n \cdot P$
$\mathbf{P_2}$	$(n+1) \cdot P$	$2(n+1) \cdot P$	...	$n(n+1) \cdot P$
...				
$\mathbf{P_m}$	$(n+1)^m \cdot P$	$2(n+1)^m \cdot P$	...	$n(n+1)^m \cdot P$

Taula 4.1: Assignació de punts als candidats

   Per tant, cada candidat i el representarem mitjançant el punt de la corba:

$$P_i = (n+1)^{i-1} \cdot P,$$

   per $i \in [\ 1, 2, .., \text{numCandidats}\ ]$

2. Cada votant genera el seu vot xifrat a partir de la suma de punts que representen als candidats escollits. Serà de la forma $[A, B] = [\ r_a \cdot P,\ \sum_{k \in C} P_k + r_a \cdot Q\ ]$ on r_a és el factor de re-xifratge del votant a i k pren el valor dels diferents elements del conjunt C, que són els candidats votats per a.

3. S'envia la suma de tots els vots xifrats:

$$Paquet = \Big[\sum_{i=1}^{nVots} A\ ,\ \sum_{i=1}^{nVots} B \Big]$$

4. Es desxifra el paquet *Paquet* i s'obté un punt de la corba R que és la suma dels punts que representen els candidats escollits pels votants.

5. Resoldre el problema de la motxilla per tal d'obtenir els resultats, o sigui, a partir d'el punt de la corba R, expressar-lo com a suma dels punts que representen a cada candidat:

$$R = t_1 \cdot P_1 + t_2 \cdot P_2 + ... + t_m \cdot P_m,$$

on P_i és el punt que representa al candidat i,

m és el nombre de candidats,

i t_i el nombre de vots aconseguits pel candidat i.

Per entendre millor tot el procés, s'exposen dos breus exemples:

Exemple 4.1 - Xifrat homomòrfic

Sigui la corba el·líptica $y^2 = x^3 + 2x + 5$ sobre $\mathbb{F}_{23}$.

Sigui $P = (16, 4)$ un generador del grup de punts de la corba.

Simulem una votació amb dos canditats i dos votants. La clau privada és $d = 17$ i la clau pública $Q = d \cdot P = 17 \cdot (16, 4) = (10, 6)$.

Els punts que representen a cada candidat són:

$$P_1 = P = (16, 4) \quad \text{i} \quad P_2 = (2 + 1) \cdot P = (1, 10).$$

Suposem la següent tria de vots:

El votant 1 vota al candidat 2.
El votant 2 vota al candidat 1.

Siguin $r_1 = 3$ i $r_2 = 2$ (aleatoris).

Els vots individuals són:

$$\text{Vot}_1 = \left[\, r_1 \cdot P \, , \;\; P_2 + r_1 \cdot Q \, \right],$$

$$\text{Vot}_1 = \left[\, 3 \cdot (16,4)\ ,\ (1,10) + 3 \cdot (10,6) \,\right] = \left[\, (1,10)\ ,\ (9,19) \,\right].$$

$$\text{Vot}_2 = \left[\, r_2 \cdot P\ ,\ P_1 + r_2 \cdot Q \,\right],$$

$$\text{Vot}_2 = \left[\, 2 \cdot (16,4)\ ,\ (16,4) + 2 \cdot (10,6) \,\right] = \left[\, (22,5)\ ,\ (22,5) \,\right].$$

Com es un xifrat homomòrfic, podem empaquetar els dos vots, i per tant enviar:

$$\text{Vots} = \left[\, (r_1 + r_2) \cdot P\ ,\ P_2 + P_1 + (r_1 + r_2) \cdot Q \,\right],$$

$$\text{Vots} = \left[\, 5 \cdot (16,4)\ ,\ (1,10) + (16,4) + 5 \cdot (10,6) \,\right],$$

$$\text{Vots} = \left[\, (12,20)\ ,\ (5,5) \,\right].$$

El resultat de desxifrar cada vot individualment serà el mateix que el de desxifrar el punt que representa la suma dels vots, i resoldre el problema de la motxilla.

Implementació de l'exemple en Sage

Listing 4.2: Exemple 4.1

```
1   # Definicio de la corba i el seu generador
2   E = EllipticCurve(GF(23), [2,5]);
3   P = E([16,4]);
4
5   # Parametres de ElGamal
6   d = 17;
7   Q = d * P;
8   r1 = 3;
9   r2 = 2;
10
11  # Punts que representen als candidats
12  p1 = gen;
13  p2 = (2+1) * gen;
14
15  # Generacio dels vots individuals
```

```
16      v1 = [r1 * P , p2 + r1 * Q];
17      v2 = [r2 * P , p1 + r2 * Q];
18
19      # Empaquetat
20      vt = [(r1+r2) * P, p2 + p1 + (r1+r2) * Q]
```

Exemple 4.2 - Xifrat i desxifrat dels vots

Sigui la corba el·líptica $y^2 = x^3 + 2x + 5$ sobre $\mathbb{F}_{23}$.

Simulem una votació amb dos canditats i tres votants. La clau privada és $d = 17$ i la clau pública $Q = 17 \cdot (16, 4) = (10, 6)$.

Els punts que representen a cada candidat són:

$$P_1 = P = (16, 4) \quad i \quad P_2 = (3 + 1) \cdot P = (8, 2).$$

Suposem la següent tria de vots:

El votant 1 vota al candidat 2.
El votant 2 vota als candidats 1 i 2.
El votant 3 vota al candidat 1.

Per tant, si empaquetem tots els vots, s'haurà d'enviar:

$$\text{Vots} = \big[\ (r_1 + r_2 + r_3) \cdot P\ ,\ P_2 + P_1 + P_1 + P_2 + (r_1 + r_2 + r_3) \cdot Q\ \big],$$

$$\text{Vots} = \big[\ 10 \cdot (16, 4)\ ,\ \ (8, 2) + (16, 4) + (12, 20) + (16, 4) + 10 \cdot (10, 6)\ \big],$$

$$\text{Vots} = \big[\ (5, 18)\ ,\ (15, 12)\ \big].$$

Un cop rebut els dos punts, desxifrem per obtenir el missatge:

$$(15, 12) - 17 \cdot (5, 18) = (5, 18).$$

Finalment, expressem el punt desxifrat en funció de suma de punts que representen als candidats (problema de la motxilla):

$$(5, 18) = (16, 4) + (16, 4) + (8, 2) + (8, 2).$$

i obtenim que els dos candidats han obtingut dos vots.

4.3 Mixing

Per tal de desvincular cada vot del seu votant, el que fem és un procés de *mixing*, és a dir, barregem els paquets que contenen els vots. El procés emula el sacsejat d'una urna en una votació tradicional. Hi ha diverses formes per barrejar els vots, però a nosaltres ens interessa que sigui universalment verificable per tal que qualsevol persona pugui comprovar la correctesa de tot el procés.

Si barregem els paquets però no fem cap altra modificació en aquests, mirant els paquets d'entrada i els de sortida seria molt senzill trobar quina associació hi ha entre ells, i per tant seria fàcil acabar relacionant-los amb els votants. Per tant, a més de barrejar els paquets, també els caldrà re-xifrar en cada node per tal d'evitar que es pugui esbrinar quina és la permutació realitzada en cada cas. El fet de treballar amb paquets de vots, enlloc de fer-ho amb els vots directament, també en dificulta aquesta associació [2]. La idea bàsica d'aquest procés queda reflectida a la figura 4.2.

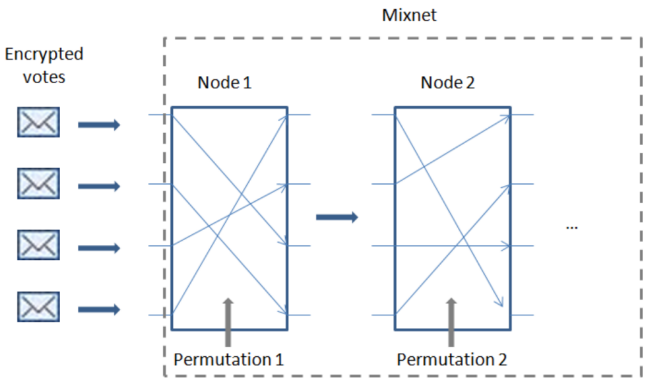

Figura 4.2: Idea bàsica del barrejat de vots

4.3.1 Random Group Full Checking

Dividim tot el procés en diverses etapes. En cadascuna d'aquestes definim un conjunt de nodes i es barregen els vots d'entrada i s'emmagatzema la permutació que s'ha dut a terme. En cada etapa es treballarà amb els vots sortints de l'etapa anterior barrejats. Com a mínim hi ha d'haver dos etapes, i a partir de quatre es considera que es disposa d'un nivell alt de seguretat.

Etapes de la mescla de vots i verificació

El mètode proposat per la barreja de vots consta dels següents pasos. Es tracta d'una adaptació del que trobem exposat a [10] per tal d'emprar-ho a nivell de paquet enlloc de fer-ho a nivell de vot i sobre corbes el·líptiques.

1. Cada mix-node barreja i re-xifra els paquets de vots rebuts (ja xifrats). S'emmagatzema quina ha estat la permutació realitzada i la suma dels factors de re-xifrat en cada cas (es repeteix aquest pas per totes les etapes).

2. Es verifica la correctesa:

   (a) Pel primer node, el verificador divideix aleatòriament els paquets d'entrada en diferents grups, seguint l'array d'agrupament que és enviat al provador.

   Sigui t el nombre de nodes, i m el nombre total de paquets, es recomana que el nombre de paquets en cada grup, n, sigui com a mínim $\sqrt[t]{m}$.

   (b) El verificador calcula la *Input Integrity Proof* (IIP) per cada node. Siguin $Pq_i = [R_i, S_i]$ per $i \in [1, n]$ els paquets d'entrada en un node. Aleshores, denotarem per:

   $$IIP = [\ E_R,\ E_S\ ]$$

   on:

$$E_R = \sum_{i=1}^{n} R_i \quad \text{i} \quad E_S = \sum_{i=1}^{n} S_i$$

al paquet format per la suma dels n paquets d'entrada.

(c) El verificador calcula la *Output Integrity Proof* (OIP) per cada node. Siguin $Pq'_i = [R'_i, S'_i]$ per $i \in [1, n]$ els paquets de sortida en un node. Aleshores, denotarem per:

$$OIP = [\ S_R,\ S_S\ ]$$

on:

$$S_R = \sum_{i=1}^{n} R'_i \quad \text{i} \quad S_S = \sum_{i=1}^{n} S'_i$$

al paquet format per la suma dels n paquets de sortida.

(d) S'ha de demostrar que OIP és un re-xifrat de IIP. Això es pot fer mitjançant una prova de coneixement nul o a partir de la suma dels valors de re-xifrat de cada paquet.

(e) Per al següent node, els grups es divideixen de forma que cada nou grup està format de vots de diferents grups de sortida del node anterior.

(f) Mentre no s'arribi a la darrera etapa, es van repetint els pasos b, c, d i e.

Aquest mètode aconsegueix bons nivells de privacitat, robustesa i solidesa.

L'algoritme per calcular la prova d'integritat dels paquets d'un node és:

Algoritme 5 Càlcul de la prova d'integritat

Entrada: llista de paquets lp, llista d'enters le (permutació)

Sortida: prova d'integritat

$P_1 \leftarrow \mathcal{O}$

$P_2 \leftarrow \mathcal{O}$

per $i = 0$ to $i <$ tamany(le) **fer**

$e \leftarrow$ le[i]

$P_1 \leftarrow P_1 + lp(e)$.getP1

$P_2 \leftarrow P_2 + lp(e)$.getP2

fi per

PI $= (P_1, P_2)$

retornar PI

Aquest algoritme és vàlid tant per calcular la *Input Integrity Proof* i la *Output Integrity Proof*. L'única diferència que hi ha entre les dues proves és els paquets que es reben a l'entrada.

4.4 Exemple del procés electoral

Simulem tot el procés d'una votació. Es van realitzant les etapes exposades en la secció 4.1 seguint l'esquema proposat en l'apartat 4.2 d'aquest mateix capítol.

Definició de la votació

Suposem una votació amb els següents paràmetres:

- **Paràmetres del sistema**:
  - Cos: $\mathbb{F}_p$ amb $p = 503$.
  - Corba $E/\mathbb{F}_p : y^2 = x^3 + 2x + 3$.

  - Cardinal de la corba $\#E(\mathbb{F}_p) = 511$ primer.
  - Un generador del grup de punts de la corba $P = (460, 313)$.
  - Clau privada: $d = 23$, clau pública: $Q = d \cdot P = 23 \cdot (460, 313) = (405, 57)$.

- **Característiques de la votació**:
  - 4 candidats.
  - Cens: 125 persones.
  - 1 vot per persona.
  - S'empaqueten 5 vots per paquet.

El primer que cal fer es calcular els punts que representaran a cada candidat:

Candidat $0 \rightarrow P_1 = P = (460, 313)$,

Candidat $1 \rightarrow P_2 = (n+1) \cdot P = 6 \cdot P = (168, 357)$,

Candidat $2 \rightarrow P_3 = (n+1)^2 \cdot P = 36 \cdot P = (300, 322)$,

Candidat $3 \rightarrow P_4 = (n+1)^3 \cdot P = 216 \cdot P = (391, 91)$.

Generació, verificació i empaquetat dels vots

Suposem que es voten els candidats que es mostren a la taula 4.2 amb els factors de xifrat que s'especifiquen. Entenem que en cada paquet s'emmagatzemen ja els 5 vots. Per tal de mostrar la informació de forma més reduïda i també més comprensible, s'afegeix la columna amb el valor del paquet un cop ja ha estat xifrat amb el criptosistema de ElGamal el·líptic. Així doncs, sigui r_i el valor aleatori de xifratge del vot i, tenim que un paquet és de la forma $[R, S]$ on:

a. Primer punt del vot xifrat:

$$R = \sum_{i=1}^{5} r_i \cdot P.$$

b. Segon punt del vot xifrat:

Sigui R_k el punt que representa el conjunt de candidats votat pel vot que ocupa la posició k dins del paquet, tenim:

$$S = R_1 + R_2 + R_3 + R_4 + R_5 + \sum_{i=1}^{5} r_i \cdot Q.$$

En la segona columna de la taula 4.2 trobem especificats quins són els cinc vots que s'empaqueten. El número que hi ha fa referència al candidat votat, és a dir, si és i el missatge d'aquell vot és P_i.

Paquet i	**Vots**	**Sumatori** r_i	**Xifrat** $(R_i,\ S_i)$
1	1, 1, 3, 2, 4	11	**(124, 346)**, **(489, 111)**
2	4, 4, 1, 2, 3	9	(448, 71), (21, 330)
3	2, 2, 2, 3, 3	8	(116, 371), (444, 412)
4	2, 3, 4, 2, 2	9	(448, 71), (75, 71)
5	2, 3, 4, 1, 1	12	(483, 71), (464, 444)
6	1, 1, 3, 2, 4	15	(489, 111), (18, 29)
7	4, 4, 1, 2, 3	13	(110, 94), (87, 104)
8	2, 2, 2, 3, 3	14	(71, 294), (189, 155)
9	2, 3, 4, 2, 2	14	(71, 294), (45, 275)
10	2, 3, 4, 1, 1	9	(448, 71), (376, 273)
11	1, 1, 3, 2, 4	9	(448, 71), (376, 273)
12	4, 4, 1, 2, 3	12	(483, 71), (407, 406)
13	2, 2, 2, 3, 3	11	(185, 69), (124, 346)
14	2, 3, 4, 2, 2	16	(17, 209), (393, 336)
15	2, 3, 4, 1, 1	8	(116, 371), (68, 489),
16	1, 1, 3, 2, 4	19	(92, 57), (212, 27)
17	4, 4, 1, 2, 3	16	(17, 209), (469, 353)
18	2, 2, 2, 3, 3	12	(483, 71), (58, 171),
19	2, 3, 4, 2, 2	10	(351, 460), (415, 294),
20	2, 3, 4, 1, 1	16	(17, 209), (287, 140),

continua ...

Paquet	**Vots**	**Sumatori** r_i	**Xifrat (R,S)**
21	1, 1, 3, 2, 4	9	$(448, 71), (376, 273)$
22	4, 4, 1, 2, 3	15	$(489, 111), (315, 162)$
23	2, 2, 2, 3, 3	11	$(124, 346), (185, 69)$
24	2, 3, 4, 2, 2	14	$(71, 294), (45, 275)$
25	2, 3, 4, 1, 1	11	$(124, 346), (489, 111)$

Taula 4.2: Paquets de vots xifrats (exemple 4.4)

A continuació, mirem com s'han obtingut els dos punts que representen al primer paquet i de la mateixa forma podem anar obtenint tota la resta:

$$R_1 = 11 \cdot P = 11 \cdot (460, 313) = (\mathbf{124}, \mathbf{346}).$$

$$S_1 = P_0 + P_0 + P_2 + P_1 + P_3 + 11 \cdot Q,$$

$$S_1 = (460, 313) + (460, 313) + (300, 322) + (168, 357) + (391, 91) + 11 \cdot (405, 57),$$

$$S_1 = (\mathbf{489}, \mathbf{111}).$$

Mixing

Per realitzar el *mixing* es divideix el procés en dues etapes de mescla i re-xifratge de paquets de vots. En cada etapa dividim el total de paquets en diversos grups. Sigui t el nombre de nodes i m el nombre total de paquets, es recomana, per motius de seguretat, que com a mínim hi hagi $n = \sqrt[t]{m}$ paquets per grup; en el nostre cas, el tamany del grup serà de $\sqrt{25} = 5$ paquets. Així doncs, com que el número de paquets és 25, per cada etapa tindrem 5 nodes amb 5 paquets d'entrada cadascun.

En la primera etapa, podem entrar els paquets aleatòriament o amb l'ordre desitjat; en l'exemple s'entren amb l'ordre amb que arriben. Cada node re-xifrarà els paquets que rep i els oferirà a la sortida barrejats.

Per fer la divisió dels paquets de sortida de la primera etapa en grups per tal de generar l'entrada de la segona etapa, només hi ha el requeriment que, en la mesura que sigui possible, els paquets d'un mateix grup han de provenir de grups diferents de l'etapa anterior. Es pot fer aquesta distribució de forma aleatòria o com en l'exemple, on el node i de la segona fase rep com a entrada els paquets que ocupen la posició i en la sortida de cadascun dels nodes de l'etapa anterior.

El sistema emmagatzema per cada node la permutació de paquets realitzada, és a dir, per cada paquet d'entrada quina és la seva posició de sortida, i la suma dels valors de re-xifratge emprats, per tal de poder comprovar, a posteriori, la correctesa de tot el procés de *mixing*. L'objectiu del mixing és desvincular un vot del votant que l'ha emès, per tant, s'ha de comprovar que, globalment, els paquets de sortida d'un node és un re-xifrat dels paquets d'entrada, per garantir que no s'ha modificat el contingut de cap vot.

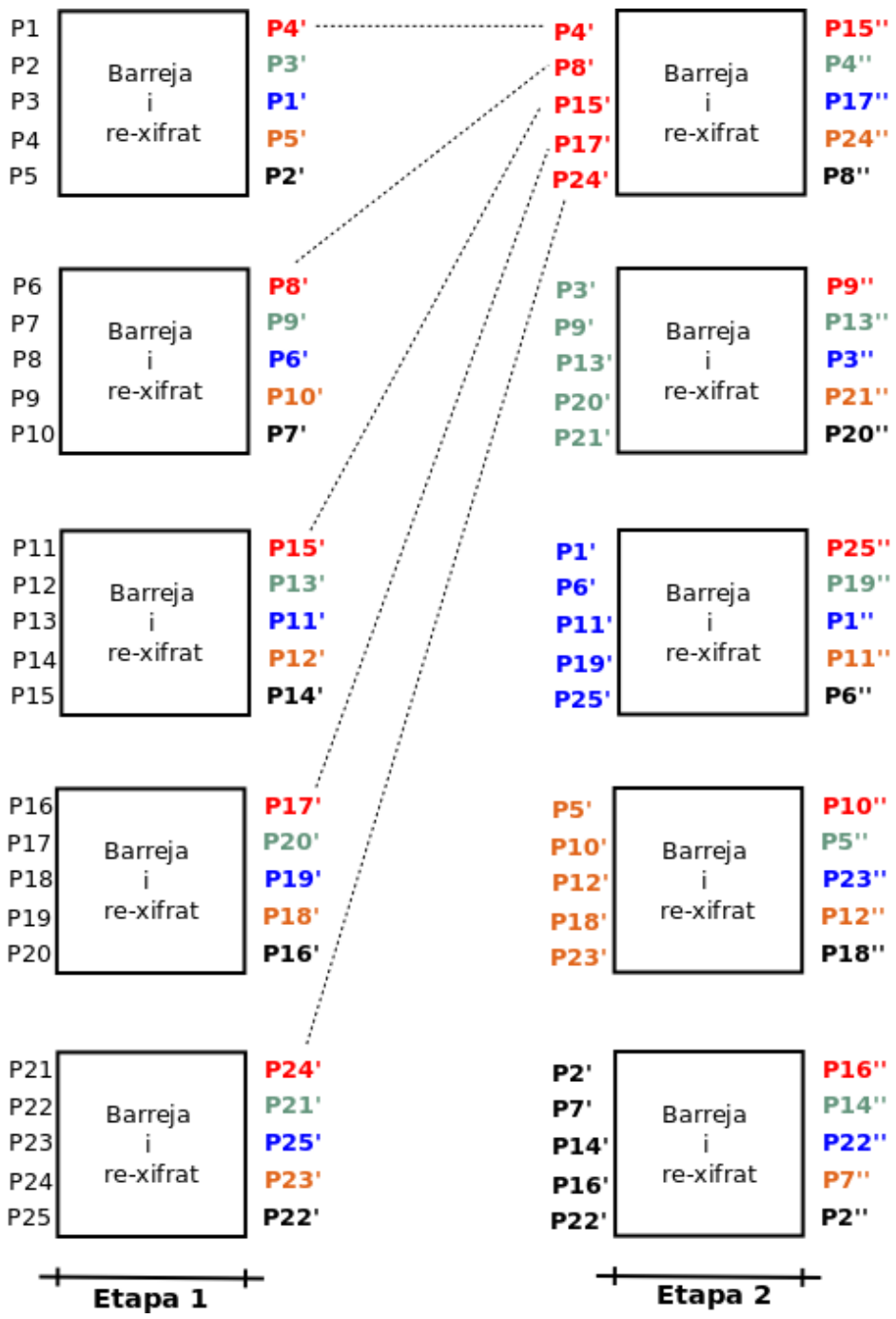

Figura 4.3: Esquema del mixing (exemple 4.4)

En cada node es re-xifren tots els paquets ja que sinó seria senzill en funció de la informació d'entrada i la de sortida anar associant els vots que tenim abans del *mixing* i els que tenim després, fins a poder relacionar el vot amb el seu votant. Els paquets i els valors de re-xifrat emprats en cada cas, es troben en la taula 4.3.

Sigui $[R, S]$ un paquet xifrat, aleshores el seu re-xifratge serà de l'estil $[R', S']$, i així

successivament.

Paq.	**Factor** ri**'**	R'_i	S'_i	**Factor** r''_i	R''_i	S''_i
1	15	(264, 125)	(428, 178)	3	(389, 149)	(221, 321)
2	7	(5, 255)	(271, 179)	20	(58, 332)	(118, 198)
3	6	(451, 159)	(482, 211)	23	(283, 24)	(123, 4)
4	18	(374, 466)	(4, 154)	17	(309, 239)	(71, 294)
5	21	(4, 154)	(134, 289)	4	(334, 115)	(109, 5)
6	2	(203, 100)	(202, 431)	21	(284, 461)	(287, 140)
7	24	(429, 496)	(224, 43)	1	(235, 328)	(381, 404)
8	11	(81, 38)	(374, 466)	4	(148, 487)	(466, 60)
9	8	(224, 460)	(311, 343)	10	(363, 40)	(207, 263)
10	18	(374, 466)	(110, 409)	24	(75, 71)	(124, 346)
11	11	(280, 188)	(193, 171)	12	(62, 376)	(116, 132)
12	20	(143, 382)	(491, 303)	16	(94, 132)	(39, 400)
13	1	(22, 249)	(20, 178)	12	(396, 121)	(264, 125)
14	5	(260, 390)	(318, 355)	18	(105, 223)	(277, 488)
15	21	(227, 355)	(221, 182)	8	(283, 24)	(129, 256)
16	17	(175, 200)	(391, 91)	3	(92, 446)	(332, 205)
17	6	(20, 325)	(394, 22)	22	(467, 121)	(44, 342)
18	19	(356, 188)	(500, 305)	1	(143, 382)	(207, 240)
19	18	(447, 278)	(193, 332)	19	(19, 484)	(180, 306)
20	18	(77, 351)	(437, 40)	22	(8, 110)	(5, 255)
21	8	(486, 51)	(405, 446)	18	(202, 431)	(373, 83)
22	19	(284, 42)	(104, 199)	10	(21, 330)	(394, 22)
23	25	(175, 303)	(287, 363)	6	(196, 185)	(55, 325)
24	9	(56, 297)	(423, 483)	3	(431, 460)	(137, 450)
25	16	(385, 497)	(376, 230)	19	(285, 229)	(25, 230)

Taula 4.3: Paquets de vots re-xifrats (exemple 4.4)

Verificació del mixing

S'ha de fer la comprovació per cada node.

Per exemple, podem comprovar el tercer node de la segona etapa.

Entrada: P'_1, P'_6, P'_{11}, P'_{19}, P'_{25}

$$E_R = (264, 125) + (203, 100) + (280, 188) + (447, 278) + (385, 497) = (483, 432)$$

$$E_S = (428, 178) + (202, 431) + (193, 171) + (193, 332) + (376, 230) = (5, 255)$$

Sortida: P''_{25}, P''_{19}, P''_1, P''_{11}, P''_6

$$S_R = (285, 229) + (19, 484) + (389, 149) + (62, 376) + (284, 461) = (41, 108)$$

$$S_S = (25, 230) + (180, 306) + (221, 321) + (116, 132) + (287, 140) = (307, 50)$$

Factors de re-xifratge: $r = r''_{25} + r''_{19} + r''_1 + r''_{11} + r''_6 = 19 + 19 + 3 + 12 + 21 = 74$

Per demostrar la correctesa de la barreja de paquets, cal comprovar:

1. $S_R = E_R + r \cdot P$

   Es compleix que: $(41, 108) = (483, 432) + 74 \cdot (405, 57)$.

2. $S_2 = E_S + r \cdot Q$

   Es compleix que: $(307, 50) = (5, 255) + 74 \cdot (460, 313)$.

aleshores el contingut emmagatzemat als paquets no ha variat.

Capítol 5

Resultats i conclusions

Per tal d'avaluar el funcionament de l'aplicació implementada es realitzen un seguit de simulacions variant els valors del cens, el nombre de candidats i el nombre de vots empaquetats en cada paquet. Les dades obtingudes queden resumides en la taula 5.1.

	65 vots / paquet					20 vots / paquet			10 vots / paquet	
Candidats / Cens	2	3	4	5	6	5	6	7	7	8
500	0,1	0,20	0,23	2,80	3,3	0,63	0,7	2,51	1,07	1,15
1000	0,18	0,40	0,45	5,58	6,52	1,27	1,37	5,05	2,12	2,27
1500	0,28	0,60	0,66	8,38	9,9	1,89	2,1	7,53	3,21	3,45
2000	0,35	0,77	0,86	10,83	13,2	2,52	2,8	10,04	4,28	4,6
2500	0,45	0,97	1,1	13,58	16,5	3,15	3,5	12,55	5,35	5,75
3000	0,53	1,17	1,33	16,48	19,8	3,78	4,2	15,06	6,42	6,9
3500	0,62	1,35	1,52	19,21	23,1	4,41	4,9	17,57	7,49	8,05
4000	0,7	1,53	1,75	21,66	26,4	5,04	5,6	20,08	8,56	9,2
5000	0,92	1,90	2,17	27,16	33	6,3	7	25,1	10,7	11,5
10000	1,78	3,80	4,32	54,32	66	12,6	14	50,2	21,4	23
15000	2,68	5,63	6,52	81,48	99	18,9	21	75,3	32,1	34,5
20000	3,57	7,50	8,68	108,64	132	25,2	28	100,4	42,8	46
25000	4,45	9,37	10,83	135,80	165	31,5	35	125,5	53,5	57,5

Taula 5.1: Temps (en segons) de les proves realitzades

La primera conclusió molt evident que podem extreure'n és que el **temps d'execució creix linealment en funció del cens**, és a dir, si no modifiquem el nombre de candidats ni la quantitat de vots que s'empaqueten, tenim que el temps és proporcional al nombre de votants.

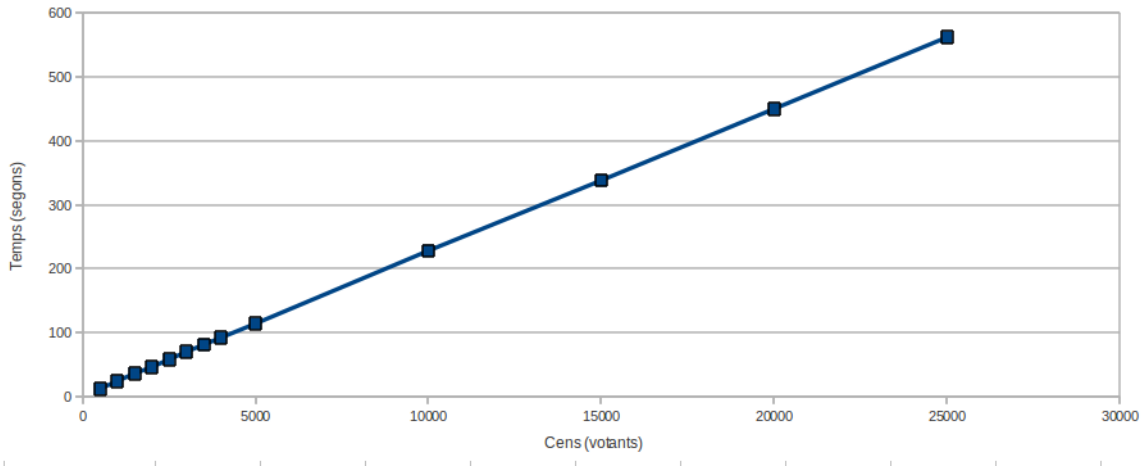

Figura 5.1: Comportament lineal en funció del cens

Si ens centrem en el *mixing* obtenim, evidentment, que a mesura que el nombre de vots per paquet augmenta el temps del barrejat de paquets disminueix ja que en total tenim menys paquets (de més vots) i tot el procés de *mixing* i la seva verificació s'ha de realitzar menys cops. Tanmateix, com ja s'ha comentat anteriorment, no podem anar emmagatzemant vots sense control, ja que el temps de desempaquetat va augmentant. Aquest fet queda clarament reflexat en la figura 5.2.

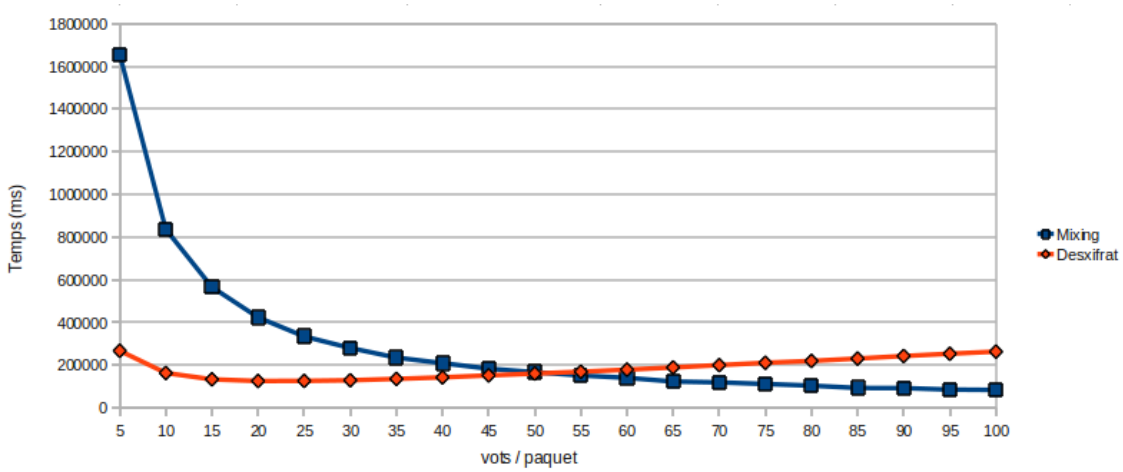

Figura 5.2: Temps de *mixing* vs temps de desempaquetat

En aquest cas, on el nombre de candidats és 5, veiem que la mida òptima dels paquets es troba entre els 45-50 vots.

Tal i com veiem a la taula 5.1 a mesura que augmentem el nombre de candidats

els temps augmenten ràpidament. Per aquest motiu, en les proves que s'han fet per estudiar el comportament del sistema el nombre de candidats no sobrepassa els 10 candidats. Com a treball futur, per millorar la implementació, una opció a tenir en compte és aplicar intel·ligència artificial. Per exemple, si en una votació es presenten 30 candidats, però sabem que la majoria dels vots recauen en dos d'aquests, es pot aprofitar aquesta informació per disminuir de forma considerable els temps de desempaquetat, provocant d'aquesta forma que es pugui augmentar el nombre de vots per paquet i d'aquesta manera el temps de *mixing* també serà més baix.

Si ho comparem amb ElGamal multiplicatiu, tenim que per aconseguir els mateixos temps amb aquest darrer, hem de treballar amb claus d'uns 600 bits, inadmissible en termes de seguretat. Així doncs, la millora d'ElGamal el·líptic respecte al multiplicatiu és important, almenys quan el nombre de candidats no és molt elevat. Per a votacions amb un número gran de candidats no s'ha fet la comparativa, ja que abans caldria implementar la millora citada anteriorment.

Apèndix A

Descripció i implementació de les proves de correctesa

En aquest apartat es descriuen i s'implementen algunes de les proves (verificacions) que s'han esmentat anteriorment.

A.1 Verificació del vot

Per cada vot s'ha de comprovar que realment representa a un candidat. Per fer-ho es realitza una prova de coneixement nul, una adaptació sobre corbes el·líptiques de la proposada a [9]. No ens preocupa molt el cost d'aquesta prova donat que es realitza a mesura que es van rebent els vots, és a dir, no s'ha d'esperar a que s'acabi el període per votar per començar a verificar els vots. Quan el servidor rep un vot, en verifica el seu contingut (que realment es voti un candidat) i l'empaqueta. Aquesta és l'única prova que es pot realitzar en temps de votació ja que per complir la propietat de no informació, ni el mixing ni el desxifratge es podran realitzar fins a l'obertura d'urnes.

Sigui $[A, B]$ el vot xifrat.

Sigui K la llista de punts que representen als m candidats diferents.

Descripció de la prova

Volem demostrar que el missatge xifrat representa realment un candidat de la llista K, és a dir, sigui T el desxifrat del vot $[A, B]$ s'ha de complir que $T \in K$:

Provador		**Verificador**
s : enter aleatori.		
$L_a = \{A_1, ..., A_m\}$: llista de punts.		
$L_b = \{B_1, ..., B_m\}$: llista de punts.		
c, w : llistes d'enters.		
Per $i \in [1, m-1]$:		
c_i, w_i aleatoris,		
$A_i = w_i \cdot P + c_i \cdot A$,		
$B_i = w_i \cdot Q + c_i(B - K_i)$.		
$A_m = s \cdot P$,		
$B_m = s \cdot Q$.		
	S'envia... L_a, L_b $\longrightarrow$	
		t: enter aleatori.
	$\longleftarrow$ Es retorna... t	
r : enter aleatori,		
$c_m = t - \sum_1^{m-1} c_i$,		
$w_m = s - c_m r$.	S'envia... c, w $\longrightarrow$	
		S'ha de comprovar:
		a. Per $i \in [1, m]$:
		$A_i = w_i \cdot P + c_i \cdot A$,
		$B_i = w_i \cdot Q + c_i \cdot (B - K_i)$.
		b. $t = \sum_1^m c_i$.

Comprovació del funcionament de la prova

Per tal de verificar el correcte comportament de la prova, s'ha de complir:

Per $i \in [1, m]$:

1. $A_i = w_i \cdot P + c_i \cdot A$

   - En el cas de $i \neq m$ és evident que el resultat és correcte. Simplement es fa la comprovació per evitar que es manipuli la llista L_a un cop s'ha rebut el valor t del verificador.
   - Per $i = m$, s'ha de comprovar que $\mathbf{s \cdot P} = w_m \cdot P + c_m \cdot A$.

$$w_m \cdot P + c_m \cdot A = (s - c_m \cdot r)P + (c - \sum_{i=1}^{m} c_i)A =$$
$$= s \cdot P - (c - \sum_{i=1}^{m} c_i) \cdot r \cdot P - (c - \sum_{i=1}^{m} c_i)A.$$

   Com que $A = r \cdot P$, per la definició de la funció de xifrat en el criptosistema de ElGamal el·líptic, tenim:

$$s \cdot P - (c - \sum_{i=1}^{m} c_i) \cdot r \cdot P - (c - \sum_{i=1}^{m} c_i) \cdot r \cdot P = \mathbf{s \cdot P}.$$

2. $B_i = w_i \cdot Q + c_i \cdot (B - K_i)$.

   - Per $i \neq m$, com en la darrera comprovació és evident que és correcte, només es fa per evitar la modificació d'algun element de la llista b.
   - Per $i = m$, s'ha de veure que $\mathbf{s \cdot Q} = w_m \cdot Q + c_m \cdot (B - K_m)$.

$$w_m \cdot Q + c_m \cdot (B - K_m) = (s - c_m \cdot r) \cdot Q + c_m \cdot (B - K_m) =$$
$$= s \cdot Q - c_m \cdot r \cdot Q + cm \cdot (B - K_m).$$

   Com que $B = r \cdot Q + K_m$, aleshores $r \cdot Q = B - K_m$ i per tant:

$$= s \cdot Q - c_m \cdot r \cdot Q + cm \cdot (B - K_m) = s \cdot Q - c_m \cdot (B - K_m) + cm \cdot (B - K_m) = \mathbf{s \cdot Q}.$$

Implementació del provador

Listing A.1: Provador de la ZKProof (validesa del vot)

```
1  public class ECZKProver implements Prover {
2      private EC curve;
3      private int nc;
4      private int nv;
5      private ArrayList<ECPoint> a, b, llistaC;
6      private ArrayList<BigInteger> c, w;
7      private BigInteger s;
8
9      public ECZKProver(EC curve, int nv, int nc){
10         this.curve = curve;
11         this.nc = nc;
12         this.nv = nv;
13         llistaC = AssignacioPunts.assignar(65, nc);
14     }
15
16     public ArrayList<ArrayList> pas1(ECElGamalBallot ballot){
17         BigInteger ci, wi;
18         ECPoint ai, bi;
19         s = BigRandom.createRandom();
20         a = new ArrayList<ECPoint>();
21         b = new ArrayList<ECPoint>();
22         c = new ArrayList<BigInteger>();
23         w = new ArrayList<BigInteger>();
24         ECPoint P = curve.getGenerator();
25         ECPoint Q = usingXML.getQ();
26         ECPoint A = ballot.getP1();
27         ECPoint B = ballot.getP2();
28
29         int i=0;
30         while(i<nc-1){
31             ci = BigRandom.createRandom();
32             wi = BigRandom.createRandom();
33             c.add(ci);
34             w.add(wi);
35             ai = curve.add(curve.multiply(wi, P),curve.multiply(ci,
                   A));
```

```
36              bi = curve.add(curve.multiply(wi, Q), curve.multiply(ci
                    , curve.substract(B, llistaC.get(i))));
37              a.add(ai);
38              b.add(bi);
39              i++;
40          }
41          ai = curve.multiply(s,P);
42          bi = curve.multiply(s, Q);
43          a.add(ai);
44          b.add(bi);
45          ArrayList<ArrayList> llistes = new ArrayList<ArrayList>();
46          llistes.add(a); llistes.add(b);
47          return llistes;
48      }
49
50      public ArrayList<ArrayList> pas3(BigInteger c, BigInteger r){
51          BigInteger ci, wi;
52          ECPoint ai, bi;
53
54          ci=c.subtract(sumaCi());
55          wi = s.subtract(ci.multiply(r));
56          this.c.add(ci);
57          w.add(wi);
58          ArrayList<ArrayList> llistes = new ArrayList<ArrayList>();
59          llistes.add(this.c); llistes.add(w);
60          return llistes;
61      }
62
63
64      private BigInteger sumaCi(){
65          BigInteger aux = BigInteger.ZERO;
66          for(int i=0; i<c.size(); i++)
67              aux = aux.add(c.get(i)).mod(curve.getOrderPrime());
68          return aux;
69      }
70
71
72  }
```

Implementació del verificador

Listing A.2: Verificador de la ZKProof (validesa del vot)

```
1  public class ECZKVerifier implements Verifier {
2
3      private EC curve;
4      private int nc;
5      private int nv;
6      private ArrayList<ECPoint> a, b, llistaC;
7      private ArrayList<BigInteger> c, w;
8      private BigInteger s;
9      private BigInteger random;
10
11     public ECZKVerifier(EC curve, int nv, int nc) {
12         this.curve = curve;
13         this.nc = nc;
14         this.nv = nv;
15         llistaC = AssignacioPunts.assignar(65, nc);
16     }
17
18     public BigInteger pas2(ArrayList<ArrayList> llistes) {
19         a = llistes.get(0);
20         b = llistes.get(1);
21         random = BigRandom.createRandom();
22         return random;
23     }
24
25     public boolean verificacioPunt(ArrayList<ArrayList> llistes,
           ECElGamalBallot ballot) {
26         boolean aux = true;
27         BigInteger suma = BigInteger.ZERO;
28         ECPoint P = this.curve.getGenerator();
29         ECPoint Q = usingXML.getQ();
30         ECPoint A = ballot.getP1();
31         ECPoint B = ballot.getP2();
32
33         c = llistes.get(0);
34         w = llistes.get(1);
35
```

```
36            for (int i = 0; i < a.size(); i++) {
37                if (!curve.add(curve.multiply(w.get(i), P), curve.
                      multiply(c.get(i), A)).equals(a.get(i))) {
38                    aux = false;
39                }
40
41                if (!curve.add(curve.multiply(w.get(i), Q), curve.
                      multiply(c.get(i), curve.substract(B, llistaC.get(i
                      )))).equals(b.get(i))) {
42                    aux = false;
43                }
44
45                suma = suma.add(c.get(i)).mod(curve.getOrderPrime());
46            }
47
48            if (!random.equals(suma)) {
49                aux = false;
50            }
51            return aux;
52        }
53    }
```

A.2 Verificació del desxifratge

És una adaptació de [11] a corbes el·líptiques.

Descripció de la prova

Sigui M el punt que representa al candidat.

Sigui $[P_1, P_2]$ el vot xifrat.

Sigui x la clau privada.

Es vol demostrar que el desxifratge s'ha fet correctament, és a dir, s'ha de comprovar que realment el punt de la corba que s'obté al desxifrar $[P_1, P_2]$ és M:

Provador		**Verificador**
s : enter aleatori, $A = s \cdot P$, $B = s \cdot P_1$, $e = hash(A\|\|B)$, $r = s + e \cdot x$.		
	$\xrightarrow{\text{S'envia... } A,\ B,\ r,\ M,\ P_1,\ P_2}$	
		$e = hash(A\|\|B)$, $R = P_2 - M$.
		Cal comprovar: $\rightarrow P \cdot r = s \cdot P + e \cdot Q$, $\rightarrow r \cdot P_1 = s \cdot P_1 + e \cdot R$.

Comprovació de funcionament de la prova

S'ha de comprovar:

1. $P \cdot r = s \cdot P + e \cdot Q$:

$$P \cdot (s + ex) = s \cdot P + e \cdot Q,$$

$$s \cdot P + ex \cdot P = s \cdot P + e \cdot Q,$$

$$ex \cdot P = e \cdot Q.$$

   Per tant, com que $Q = x \cdot P$ es compleix que $e \cdot Q = ex \cdot P$.

2. $r \cdot P_1 = s \cdot P_1 + e \cdot R$:

$$(s + ex) \cdot P_1 = s \cdot P_1 + e \cdot (P_2 - M),$$

$$s \cdot P_1 + ex \cdot P_1 = s \cdot P_1 + e(P_2 - M),$$

$$ex \cdot P_1 = e \cdot (P_2 - M) \rightarrow x \cdot P_1 = P_2 - M.$$

   Per la definició de la funció de xifrat de ElGamal el·líptic, tenim que $P_1 = k \cdot P$ i $P_2 = M + k \cdot Q$, per tant

$$xK \cdot P = M + k \cdot Q - M,$$

$$xk \cdot P = k \cdot Q.$$

   com que $x \cdot P = Q$ és compleix la igualtat anterior.

Implementació del provador

Listing A.3: Provador de la prova del desxifrat

```
1  public class ECCPProver implements Prover{
2      protected EC curve;
3
4      public ECCPProver(EC c2){
5          curve=new EC(c2);
6      }
7
8      public void prover(ECPoint m,  ECElGamalBallot eg){
9          BigInteger x=usingXML.loadPrivKey();
10         ECPoint c=eg.getP1();
11         ECPoint d=eg.getP2();
12
13         // Generate random s
14         Random rnd=new SecureRandom();
15         BigInteger lim = curve.getOrderPrime();
16         BigInteger s = new BigInteger(lim.bitLength(), rnd).mod(
               curve.getOrderPrime());
17
18         // Computes a and b
19         ECPoint A=curve.multiply(s, curve.getGenerator());
20         ECPoint B=curve.multiply(s, c);
21
22         // Compute e = hash(a concatenate b)
23         String con=curve.pointToString(A).concat(curve.
               pointToString(B));
24         long e=con.hashCode();
25
26         // Compute r = s+e*x
27         BigInteger r=(s.add(x.multiply(BigInteger.valueOf(e)))).mod
               (curve.getOrderPrime());
28
29         // Creacio d'un fitxer XML amb (A,B,r,m,c,d)
30         usingXML.saveABrmcd(A, B, r, m, c, d);
31     }
32 }
```

Implementació del verificador

Listing A.4: Verificador de la prova del desxifrat

```
1  public class ECCPVerifier implements Verifier{
2      protected EC curve;
3      protected ECPoint Q; //x*P
4
5      public ECCPVerifier(){
6          // Llegir del XML
7          curve=usingXML.getCurve();
8          Q=usingXML.getQ();
9      }
10
11     public boolean verifier(){
12         ECPoint A, B, m, c, d;
13         BigInteger r;
14
15         // llegeix a,b,r,m,c,d del fitxer XML
16         ...
17
18         // Compute e = hash(a concatenate b)
19         String con=curve.pointToString(A).concat(curve.
               pointToString(B));
20         long e=con.hashCode();
21
22         ECPoint dSm=curve.substract(d, m); // Compute v = d - m
23
24         // Check g*r = a+y*e
25         if(!(curve.multiply(r, curve.getGenerator())).equals(curve.
               add(A, curve.multiply(BigInteger.valueOf(e), Q))))
               return false;
26
27         // Check u*r = b+v*e on u=c i v=d-m
28         if(!(curve.multiply(r, c)).equals(curve.add(B, curve.
               multiply(BigInteger.valueOf(e), dSm)))) return false;
29
30         return true;
31     }
32 }
```

Apèndix B

Implementació del sistema de votació electrònica

En aquest apèndix es mostra el codi d'algunes de les funcions del sistema de votació electrònica implementat.

B.1 Implementació de les funcionalitats rellevants

Les operacions de les quals es mostra el codi són:

- Generació d'un vot xifrat aleatori.
- Empaquetat d'un vot
- Desxifrat i desempaquetat d'un vot

Generació i encriptació d'un vot

Listing B.1: Generació d'un vot xifrat

```
1  public ECElGamalBallot randVote(){
2      ECElGamalBallot vot= new ECElGamalBallot();
3
4      // Eleccio del candidat (aleatori)
```

```
5       Random generator = new Random();
6       int a = generator.nextInt(num_candidats);
7
8       // Es xifra el Vot
9       BigInteger r =  BigRandom.createRandom();
10      vot=cy.encrypt(PesCandidat.get(a),r);
11      return vot;
12  }
13
14  ...
15
16  public ECElGamalBallot encrypt(ECPoint p, BigInteger r){
17      ECPoint p1 = c.multiply(r, c.getGenerator());
18      ECPoint p2 = c.add(p, c.multiply(r, this.Q));
19
20      return new ECElGamalBallot(p1, p2);
21   }
```

Empaquetat d'un vot

Listing B.2: Empaquetat del vots

```
1    public void getVote(ECElGamalBallot vote) {
2        if (lPaquets.get(lPaquets.size() - 1).isFull()) {
3            lPaquets.add(new ECElGamalPackage(this.maxVotsPack));
4        }
5        lPaquets.get(lPaquets.size() - 1).addBallot(vote);
6    }
```

Desempaquetat i desxifrat de vots

Listing B.3: Desxifrat dels vots

```
1  public Integer[] unpack(Cryptosystem CS, boolean CP, int
       num_candidats) {
2
3        ECPoint m = (ECPoint) CS.Decrypt(pack);
4
5        //Chaum-Pedersen's proof
6        if (CP == true) {
7            ECCPProver CPprover = new ECCPProver(c);
```

```
8             ECCPVerifier CPverifier = new ECCPVerifier();
9             CPprover.prover(m, pack);
10            if (CPverifier.verifier()) {
11                ECKnapsackMITM motxilla = new ECKnapsackMITM(c, m,
                      num_candidats, capacity);
12                return (Integer[]) motxilla.solveAlgorithm();
13            } else {
14                return null;
15            }
16        } else {
17            ECKnapsackMITM motxilla = new ECKnapsackMITM(c, m,
                  num_candidats, capacity);
18            return (Integer[]) motxilla.solveAlgorithm();
19        }
20
21    }
```

B.2 Problema de la motxilla

El problema de la motxilla consisteix en donat un punt de la corba T que és suma d'uns quants punts que representen als diferents candidats, trobar aquests sumands per tal de poder fer el recompte de vots.

Sigui $K = \{P_1, P_2, ..., P_{numCandidats}\}$ el conjunt de punts que representen a cada candidat.

Aleshores, s'ha de trobar quins punts del conjunt K sumats donen T. [12]

Listing B.4: Algoritme de la motxilla

```
public Object solveAlgorithm() {
2       /*
3        * Knapsack Meet in the middle
4        * cost = O(n*2^n/2)
5        */
6       Integer solution[] = new Integer[num_candidats];
7       Integer combination[] = new Integer[num_candidats-(
            num_candidats/2)];
8       Integer combination2[] = new Integer[(num_candidats/2)];
9       Boolean last_combination = false;
10      HashMap table = new HashMap(maxVotesCandidate^combination.
            length);
11
12      // Primera combinacio
13      for (int i = 0; i < combination.length; i++) {
14          combination[i] = 0;
15      }
16      for (int i = 0; i < combination2.length; i++) {
17          combination2[i] = 0;
18      }
19
20      // Posem les primeres combinacions en un hashmap
21      actual=ECPoint.POINT_INFINITY;
22
23      while (!actual.equals(Weight) && !last_combination) {
24          table.put(curve.substract(Weight, actual), combination.
```

```
                clone());
25          last_combination=this.nextCombination(combination,curve.
                getGenerator());
26      }
27
28      if (!last_combination) {
29          //Si ja trobem la solucio la retormem amb segona combinacio
                tot a 0
30          System.arraycopy(combination, 0, solution, 0, combination.
                length);
31          System.arraycopy(combination2, 0, solution, combination.
                length, combination2.length);
32          return solution;
33      } else {
34          // Provem les segones combinacions i les busquem al HashMap
35          last_combination = false;
36          actual=ECPoint.POINT_INFINITY;
37          ECPoint jump = curve.multiply((int)Math.pow(
                maxVotesCandidate+1,combination.length), curve.
                getGenerator());
38          while (!table.containsKey(actual) && !last_combination) {
39              last_combination=nextCombination(combination2,jump);
40          }
41          if(!last_combination){
42              System.arraycopy(table.get(actual), 0, solution, 0,
                    combination.length);
43              System.arraycopy(combination2, 0, solution, combination
                    .length, combination2.length);
44              return solution;
45          }else{
46              // No troba solucio
47              return null;
48          }
49      }
50  }
```

Bibliografia

[1] I. Blake, G. Seroussi, N. Smart. "Elliptic Curves in Cryptography". London Mathematical Society, LNS 265, Cambridge University Press, 2000.

[2] S. Bosch, "Efficient Cryptosystem for universally veriable mixnets". Treball final de carrera. Universitat Politècnica de Catalunya, 2009.

[3] N. Busom, "Disseny d'una plataforma de votació electrònica: mòdul criptogràfic". Treball final de carrera. Universistat de Lleida, 2010.

[4] V. Mateu, "Votació electrònica amb recompte homomòrfic". Treball final de carrera. Universitat de Lleida, 2010.

[5] V. Mateu,"Implementació d'un sistema de votació sobre la xifra de Paillier i ElGamal". Treball final de màster. Universitat de Lleida, 2009.

[6] A.J. Menezes, P. van Oorschot, S.A. Vanstone. "The handbook of Applied Cryptography. CRC Press, 1997.

[7] J.M. Miret, R. Moreno, J. Pujolàs, M. Valls, "Algorithms and cryptographic protocols using elliptic curves". Contributions to science, vol 3, 481-491. Universitat de Lleida, 2007.

[8] V.M. Morales, "Seguridad en los procesos de voto electrónico remoto: registro, votación, consolidación de resultados y auditoría". Tesi doctoral. Universitat Politècnica de Catalunya, 2009.

[9] K. Peng, "A hybrid E-voting scheme". Lecture Notes in Computer Science, vol. 5451, 2009.

[10] J. Puiggalí , S. Guasch,"Universally Verifiable Efficient Re-encryption Mixnet". 4th International Conference on Electronic Voting (EVOTE '10), 2010.

[11] F. Sebé, J.M. Miret, J. Pujolàs, J. Puiggalí, "Simple and efficient hash-based verifiable mixing for remote electronic voting". Computer Communications, vol 33, num 6, 2010.

[12] S. Sisó, "An electronic voting platform with elliptic curve criptography". Treball final de carrera. Universitat de Lleida, 2011.

[13] W. Stein, D. Joyner, "SAGE: System for Algebra and Geometry Experimentation", http://www.sagemath.org, 2005.